UNIDAD 2
ACTIVIDAD PRÁCTICA
ALGORITMOS Y ESTRUCTURAS DE DATOS

U-tad
CENTRO UNIVERSITARIO DE TECNOLOGÍA Y ARTE DIGITAL

Centro adscrito a
Universidad Camilo José Cela

ÍNDICE

ORGANIZACIÓN DE LA ACTIVIDAD PRÁCTICA

Nombre de la práctica	
Tipo de tarea	Individual
Entregables	Archivo zip con los diferentes ejercicios del alumno (Revisar apartado: "Como realizar mi entrega")

DESCRIPCIÓN DE LA ACTIVIDAD PRÁCTICA

- Ejercicio 1 (3 puntos)

Escribe un programa que actúe como un conversor de pulgadas a centímetros y viceversa desde la línea de comandos. El programa permitirá al usuario ingresar un valor junto con la unidad de origen y la unidad de destino, y luego realizará la conversión correspondiente.

El programa deberá aceptar tres argumentos de línea de comandos:

- Valor númérico que se quiere convertir en formato coma flotante.
- Una palabra que representa la unidad de origen: puede ser "inch" ó "cm"
- Una palabra que representa la unidad de destino: puede ser "inch" ó "cm"

Se pide implementar una función que convierta los datos de entrada a las unidades pedidas:

- double convertir_longitud(ConversionInfo_t conversion);

  La función recibirá una estructura que contenga información de los datos, el tipo de datos de entrada (cm ó pulgadas) y tipo de datos de salida (cm ó pulgadas) pedidos.

Se valorará la calidad de la implementación en base a las siguientes pautas:

- Se debe guardar en una estructura tanto el tipo de unidad como el valor de origen y destino.
- Se valorará el uso correcto de enumerados para esta implementación.
- Se valorará el control de errores. Si el número introducido o si las unidades especificadas no son válidas o no se reconocen, el programa deberá mostrar un mensaje de error apropiado y terminar la ejecución.
- Se valorará el uso de funciones para pasar valores "cadena" a valores "enumerado" para realizar las operaciones pedidas.

Una pulgada equivale a 2.54 centímetros.

Ejemplo de ejecución:

```
$ ./conversor.exe 100 inch cm
100.00 inch es equivalente a 254.00 cm
```

- Ejercicio 2 (7 puntos)

Simular una batalla entre dos gladiadores, uno será el usuario y el otro la máquina.
Cada gladiador tiene un nombre (hasta 10 caracteres), un tipo de arma, una fuerza y una salud.
Los ataques dependerán de la fuerza del gladiador y del tipo de arma, restarán salud a su contrincante. Deberá haber dos tipos de arma:

- Espada: el ataque será la fuerza del gladiador.
- Lanza: en este caso será el doble de la fuerza del gladiador.

El programa deberá permitir al usuario ingresar los nombres de los dos gladiadores y datos de juego desde la terminal. Al ejecutar el programa se le pasarán los datos de dos gladiadores (usuario y cpu en ese orden). Deberá pasarle los siguientes parámetros (Espartaco será el usuario, Marcus la máquina):

./gladiadores.exe Espartaco,20,espada Marcus,10,lanza

Al inicio del programa, ambos jugadores tienen una salud de 100 unidades. Durante la batalla, el usuario tendrá la oportunidad de atacar o defenderse en cada turno mediante un menú. En el caso de **la máquina atacará con una probabilidad del 50% aleatorio**.

La batalla se desarrollará por turnos **hasta que uno de los gladiadores quede sin salud.**
Al finalizar la batalla, el programa deberá mostrar el resultado y mostrar los datos del ganador de la batalla.
Ejemplo de ejecución:

```
$ ./gladiadores.exe Espartaco,20,espada Marcus,10,lanza
Turno de Espartaco:
1. Atacar
2. Defender
Selecciona una opcion: 1
Atacaste a Marcus y le hiciste 20 de danio!
Turno de Marcus:
Marcus te ataca y te hace 20 de danio!
Turno de Espartaco:
1. Atacar
2. Defender
Selecciona una opcion: 2
Te has defendido.
```

```
Has ganado la batalla!
Nombre: Espartaco
Tipo de arma: Espada
Fuerza: 20
Salud: 20
```

RECOMENDACIONES E INDICACIONES PARA LA ENTREGA

- Claridad en el código:

Asegúrate de que tu código sea claro y esté bien estructurado. Sigue las buenas prácticas de programación, como el uso de indentaciones y comentarios, para facilitar su lectura y comprensión.

- Uso de identificadores descriptivos:

En la implementación en C, utiliza nombres de variables que describan claramente su propósito (por ejemplo, exponente en lugar de solo exp).

- Revisa los datos de entrada:

Antes de implementar los algoritmos en C, verifica que tienes claro cuáles son los posibles datos que un usuario puede introducir en tu programa y asegúrate de que no da errores al introducir datos no admitidas como letras cuando se soliciten números.

RÚBRICA DE CORRECCIÓN

La rúbrica para corregir el ejercicio seguirá los criterios listados a continuación, que tienen distintos pesos respecto al total de la nota.

Criterios	Excelente	Satisfactorio	No satisfactorio	Insuficiente
Ejercicio 1: Conversor	De 3 a 2.25 puntos: Programa funcional y correcto, con buen uso de condicionales y bucles y funciones, estructuras, argc/argv, lógica clara y sin errores importantes. Y revisión de parámetros de entrada.	De 2.24 a 1.5 puntos: Programa funcional con algunos errores menores en la lógica o en la implementación, pero produce resultados correctos.	De 1.49 a 0.75 punto: Programa con errores significativos que no produce el resultado correcto, aunque hay un intento de implementación.	De 0.74 a 0 puntos: Programa incorrecto o muy incompleto, sin funcionalidad significativa.
Ejercicio 2: Gladiadores	De 7 a 5.25 puntos: Programa funcional y correcto, con buen uso de condicionales y bucles y funciones, estructuras, argc/argv, lógica clara y sin errores importantes. Y revisión de	De 5.24 a 3.5 puntos: Programa funcional con algunos errores menores en la lógica o en la implementación, pero produce resultados correctos.	De 3.49 a 1.75 punto: Programa con errores significativos que no produce el resultado correcto, aunque hay un intento de implementación.	De 1.74 a 0 puntos: Programa incorrecto o muy incompleto, sin funcionalidad significativa.

	parámetros de entrada.			

COMO REALIZAR LA ENTREGA

Para realizar le entrega de esta práctica se pide al alumno entregas un archivo zip con el nombre de NombreAlumno_PrimerApellido_PracticaTema2.zip.

El zip debe contener los siguientes archivos:

- Un archivo llamado ejercicio1.c. Este archivo debe contener el código compilable del ejercicio 1.
- Un archivo llamado ejercicio2.c. Este archivo debe contener el código compilable del ejercicio 2

Además, se solicita al alumno que todos los archivos vayan iniciados con un comentario multilínea con indicando su nombre y el número del ejercicio.